كلية العلوم و التقنيات فاس
ⵜⴰⵖⵙⵓⵍⵜ ⵏ ⵜⵓⵙⵙⵏⵉ ⴷ ⵜⵓⵏⵣⵣⴰⵜ
Faculté des Sciences et Techniques de Fès

جامعة سيدي محمد بن عبد الله
ⵜⴰⵙⴷⴰⵡⵜ ⵙⵉⴷⵉ ⵎⵓⵃⵎⴷ ⴱⵏ ⵄⴱⴷⵍⵍⴰ
Université Sidi Mohamed Ben Abdellah

TP1: ACP

Réalisé par:

- El Ghoul Abdessalam

Master – S.I.R
2021/2022

TP 1 : ACP

- ## Principe de ACP :

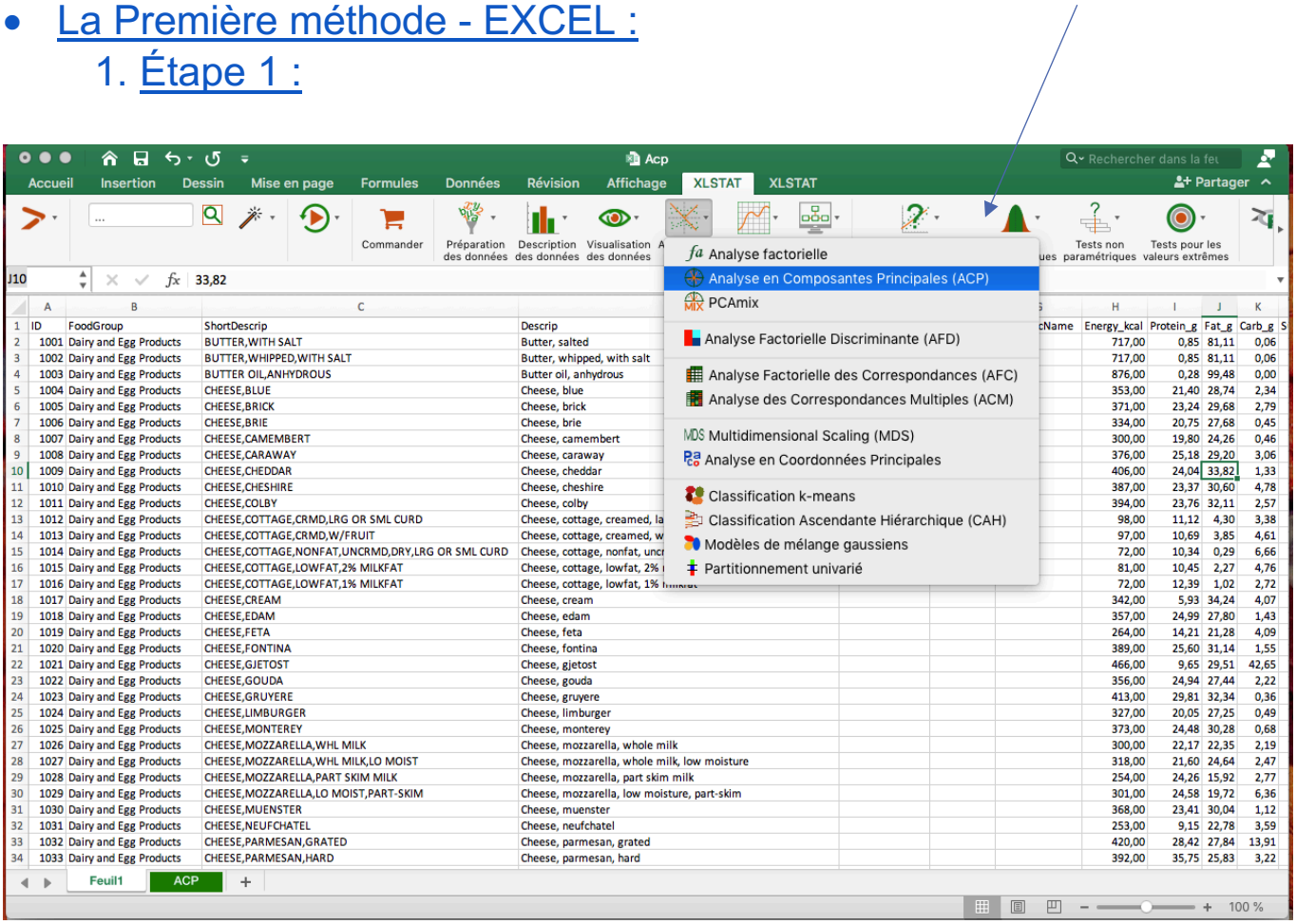
ACP est une méthode de la famille de l'analyse des données et plus généralement de la statistique multivariée, qui consiste à transformer des variables liées entre elles (dites « corrélées » en statistique) en nouvelles variables décorrélées les unes des autres. Ces nouvelles variables sont nommées « composantes principales », ou axes principaux. Elle permet au praticien de réduire le nombre de variables et de rendre l'information moins redondante.

- ## Data set utiliser :

La base de donnée utiliser est une version aplatie de la base de données nationale des éléments nutritifs de l'USDA. Il est extrait de la base de données MongoDB créée dans un projet

Lien de base de données : Lien

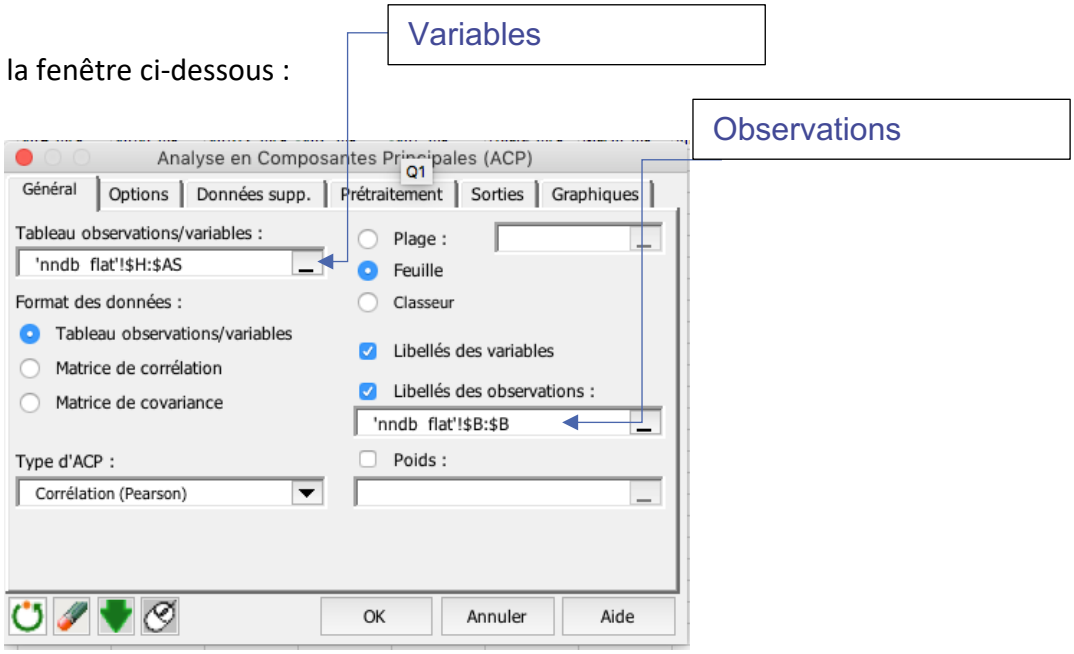
- ## La Première méthode - EXCEL :
  1. ### Étape 1 :



	ID	FoodGroup	ShortDescrip	Descrip	Energy_kcal	Protein_g	Fat_g	Carb_g
2	1001	Dairy and Egg Products	BUTTER,WITH SALT	Butter, salted	717,00	0,85	81,11	0,06
3	1002	Dairy and Egg Products	BUTTER,WHIPPED,WITH SALT	Butter, whipped, with salt	717,00	0,85	81,11	0,06
4	1003	Dairy and Egg Products	BUTTER OIL,ANHYDROUS	Butter oil, anhydrous	876,00	0,28	99,48	0,00
5	1004	Dairy and Egg Products	CHEESE,BLUE	Cheese, blue	353,00	21,40	28,74	2,34
6	1005	Dairy and Egg Products	CHEESE,BRICK	Cheese, brick	371,00	23,24	29,68	2,79
7	1006	Dairy and Egg Products	CHEESE,BRIE	Cheese, brie	334,00	20,75	27,68	0,45
8	1007	Dairy and Egg Products	CHEESE,CAMEMBERT	Cheese, camembert	300,00	19,80	24,26	0,46
9	1008	Dairy and Egg Products	CHEESE,CARAWAY	Cheese, caraway	376,00	25,18	29,20	3,06
10	1009	Dairy and Egg Products	CHEESE,CHEDDAR	Cheese, cheddar	406,00	24,04	33,82	1,33
11	1010	Dairy and Egg Products	CHEESE,CHESHIRE	Cheese, cheshire	387,00	23,37	30,60	4,78
12	1011	Dairy and Egg Products	CHEESE,COLBY	Cheese, colby	394,00	23,76	32,11	2,57
13	1012	Dairy and Egg Products	CHEESE,COTTAGE,CRMD,LRG OR SML CURD	Cheese, cottage, creamed, la	98,00	11,12	4,30	3,38
14	1013	Dairy and Egg Products	CHEESE,COTTAGE,CRMD,W/FRUIT	Cheese, cottage, creamed, w	97,00	10,69	3,85	4,61
15	1014	Dairy and Egg Products	CHEESE,COTTAGE,NONFAT,UNCRMD,DRY,LRG OR SML CURD	Cheese, cottage, nonfat, uncr	72,00	10,34	0,29	6,66
16	1015	Dairy and Egg Products	CHEESE,COTTAGE,LOWFAT,2% MILKFAT	Cheese, cottage, lowfat, 2%	81,00	10,45	2,27	4,76
17	1016	Dairy and Egg Products	CHEESE,COTTAGE,LOWFAT,1% MILKFAT	Cheese, cottage, lowfat, 1%	72,00	12,39	1,02	2,72
18	1017	Dairy and Egg Products	CHEESE,CREAM	Cheese, cream	342,00	5,93	34,24	4,07
19	1018	Dairy and Egg Products	CHEESE,EDAM	Cheese, edam	357,00	24,99	27,80	1,43
20	1019	Dairy and Egg Products	CHEESE,FETA	Cheese, feta	264,00	14,21	21,28	4,09
21	1020	Dairy and Egg Products	CHEESE,FONTINA	Cheese, fontina	389,00	25,60	31,14	1,55
22	1021	Dairy and Egg Products	CHEESE,GJETOST	Cheese, gjetost	466,00	9,65	29,51	42,65
23	1022	Dairy and Egg Products	CHEESE,GOUDA	Cheese, gouda	356,00	24,94	27,44	2,22
24	1023	Dairy and Egg Products	CHEESE,GRUYERE	Cheese, gruyere	413,00	29,81	32,34	0,36
25	1024	Dairy and Egg Products	CHEESE,LIMBURGER	Cheese, limburger	327,00	20,05	27,25	0,49
26	1025	Dairy and Egg Products	CHEESE,MONTEREY	Cheese, monterey	373,00	24,48	30,28	0,68
27	1026	Dairy and Egg Products	CHEESE,MOZZARELLA,WHL MILK	Cheese, mozzarella, whole milk	300,00	22,17	22,35	2,19
28	1027	Dairy and Egg Products	CHEESE,MOZZARELLA,WHL MILK,LO MOIST	Cheese, mozzarella, whole milk, low moisture	318,00	21,60	24,64	2,47
29	1028	Dairy and Egg Products	CHEESE,MOZZARELLA,PART SKIM MILK	Cheese, mozzarella, part skim milk	254,00	24,26	15,92	2,77
30	1029	Dairy and Egg Products	CHEESE,MOZZARELLA,LO MOIST,PART-SKIM	Cheese, mozzarella, low moisture, part-skim	301,00	24,58	19,72	6,36
31	1030	Dairy and Egg Products	CHEESE,MUENSTER	Cheese, muenster	368,00	23,41	30,04	1,12
32	1031	Dairy and Egg Products	CHEESE,NEUFCHATEL	Cheese, neufchatel	253,00	9,15	22,78	3,59
33	1032	Dairy and Egg Products	CHEESE,PARMESAN,GRATED	Cheese, parmesan, grated	420,00	28,42	27,84	13,91
34	1033	Dairy and Egg Products	CHEESE,PARMESAN,HARD	Cheese, parmesan, hard	392,00	35,75	25,83	3,22

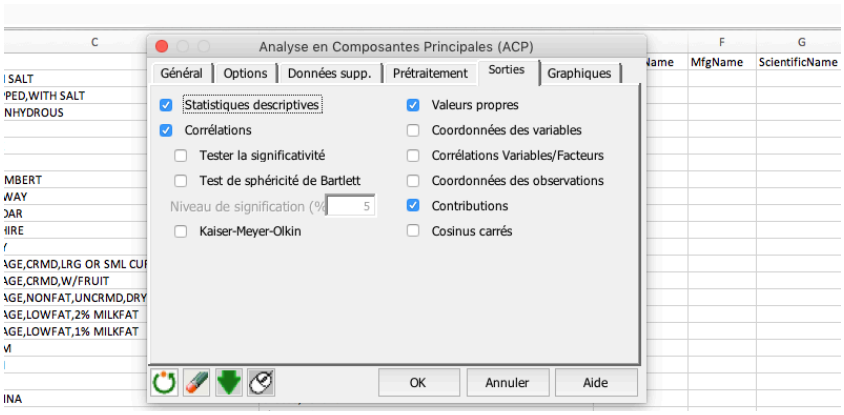
2. Étape 2 :

On remplit les champs de la fenêtre ci-dessous :



3. Étape 3 :

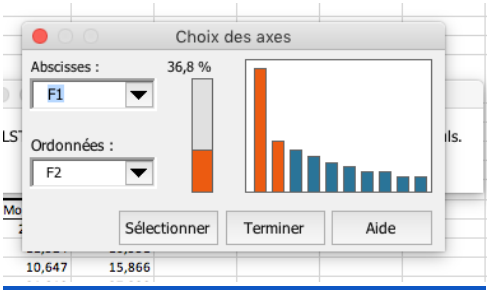
On remplit les champs de la fenêtre(outputs) ci-dessous :



4. Résultats:

- confirmer les axes ACP pour lesquels vous souhaitez afficher des graphiques 2D.

- Dans cet exemple, le pourcentage de variabilité représenté par les deux premiers facteurs n'est pas très élevé (36,8 %).



- Une table contenant quelques informations sur les variables : moyenne , maximum , min…

Statistiques descriptives :

Variable	Observations	données ma	données ma	Minimum	Maximum	Moyenne	Ecart-type
Energy_kcal	8618	0	8618	0,000	902,000	226,439	169,389
Protein_g	8618	0	8618	0,000	88,320	11,524	10,551
Fat_g	8618	0	8618	0,000	100,000	10,647	15,866
Carb_g	8618	0	8618	0,000	100,000	21,819	27,239
Sugar_g	8618	0	8618	0,000	99,800	6,560	13,602
Fiber_g	8618	0	8618	0,000	79,000	2,023	4,314
VitA_mcg	8618	0	8618	0,000	30000,000	93,969	779,362
VitB6_mg	8618	0	8618	0,000	6667,000	24,223	246,966
VitB12_mcg	8618	0	8618	0,000	98,890	1,225	4,319
VitC_mg	8618	0	8618	0,000	2400,000	7,925	57,583
VitE_mg	8618	0	8618	0,000	149,400	0,872	3,849
Folate_mcg	8618	0	8618	0,000	5881,000	50,306	186,561
Niacin_mg	8618	0	8618	0,000	90567,000	1717,581	3408,344
Riboflavin_m	8618	0	8618	0,000	3846,000	22,258	212,696
Thiamin_mg	8618	0	8618	0,000	23375,000	21,727	323,576
Calcium_mg	8618	0	8618	0,000	7364,000	73,411	201,364
Copper_mcg	8618	0	8618	0,000	14588,000	34,895	391,583
Iron_mg	8618	0	8618	0,000	123,600	2,697	5,727
Magnesium_	8618	0	8618	0,000	781,000	32,745	56,068

- la matrice de corrélation.

Matrice de corrélation (Pearson (n)) :

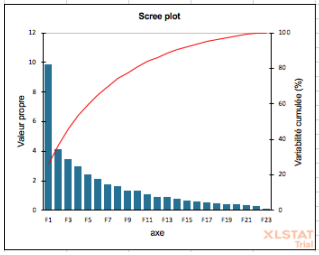
Variables	Energy_kcal	Protein_g	Fat_g	Carb_g	Sugar_g	Fiber_g	VitA_mcg	VitB6_mg	VitB12_mcg	VitC_mg	VitE_mg	Folate_mcg	Niacin_mg	Riboflavin_mg	Thiamin_mg	Calcium_mg	Copp
Energy_kcal	1	0,110	0,807	0,487	0,313	0,197	0,026	0,122	-0,012	-0,033	0,304	0,146	0,175	0,156	0,187	0,124	
Protein_g	0,110	1	0,055	-0,302	-0,266	-0,073	0,026	0,228	0,245	-0,066	-0,029	0,009	0,377	0,202	0,098	0,047	
Fat_g	0,807	0,055	1	-0,054	-0,002	-0,029	0,024	-0,047	-0,021	-0,060	0,338	-0,058	-0,023	-0,036	-0,006	0,015	
Carb_g	0,487	-0,302	-0,054	1	0,615	0,458	0,002	0,196	-0,095	0,077	0,070	0,325	0,173	0,232	0,282	0,189	
Sugar_g	0,313	-0,266	-0,002	0,615	1	0,116	0,010	0,089	-0,050	0,064	0,066	0,161	0,034	0,125	0,098	0,130	
Fiber_g	0,197	-0,073	-0,029	0,458	0,116	1	0,006	0,243	-0,052	0,082	0,160	0,249	0,135	0,158	0,205	0,232	
VitA_mcg	0,026	0,026	0,024	0,002	0,010	0,006	1	0,137	0,581	0,111	0,044	0,109	0,152	0,326	0,053	0,035	
VitB6_mg	0,122	0,228	-0,047	0,196	0,089	0,243	0,137	1	0,264	0,271	0,286	0,550	0,714	0,582	0,417	0,177	
VitB12_mcg	-0,012	0,245	-0,021	-0,095	-0,050	-0,052	0,581	0,264	1	0,007	0,061	0,158	0,279	0,436	0,100	0,012	
VitC_mg	-0,033	-0,066	-0,060	0,077	0,064	0,082	0,111	0,271	0,007	1	0,067	0,073	0,209	0,226	0,057	0,112	
VitE_mg	0,304	-0,029	0,338	0,070	0,066	0,160	0,044	0,286	0,061	0,067	1	0,184	0,197	0,157	0,119	0,110	
Folate_mcg	0,146	0,009	-0,058	0,325	0,161	0,249	0,109	0,550	0,158	0,073	0,184	1	0,535	0,571	0,539	0,149	
Niacin_mg	0,175	0,377	-0,023	0,173	0,034	0,135	0,152	0,714	0,279	0,209	0,197	0,535	1	0,747	0,602	0,133	
Riboflavin_m	0,156	0,202	-0,036	0,232	0,125	0,158	0,326	0,582	0,436	0,226	0,157	0,571	0,747	1	0,634	0,222	
Thiamin_mg	0,187	0,098	-0,006	0,282	0,098	0,205	0,053	0,417	0,100	0,057	0,119	0,539	0,602	0,634	1	0,136	
Calcium_mg	0,124	0,047	0,015	0,189	0,130	0,232	0,035	0,177	0,012	0,112	0,110	0,149	0,133	0,222	0,136	1	

- Les valeurs propres, qui reflètent la qualité de la projection du tableau initial à 38 dimensions vers un nombre inférieur de dimensions.

Analyse en Composantes Principales :

Valeurs propres :

	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15	F16
Valeur propre	9,869	4,115	3,427	2,932	2,381	2,093	1,743	1,628	1,326	1,322	1,035	0,906	0,876	0,784	0,652	0,572
Variabilité (%	25,970	10,828	9,019	7,717	6,267	5,509	4,586	4,284	3,490	3,478	2,724	2,384	2,305	2,064	1,715	1,504
% cumulé	25,970	36,798	45,817	53,534	59,801	65,310	69,896	74,179	77,669	81,147	83,871	86,256	88,561	90,625	92,340	93,844



On a trouvé que la variabilité cumule pour les 10 premiers composants égale a 81.147%.

- La Deuxième méthode - Python :
  1. Importation de librairies :

```
# pandas : panel data , pour une facile manipulation des données.
import pandas as pd
# numpy : numerical python extensions , pour la manipulation de tableaux et les opérations mathématiques.
import numpy as np
# decomposition : Package pour l'analyse en composantes principales de scikit learn.
from sklearn.decomposition import PCA
# preprocessing : package pour centrer et normaliser les données avant d'effectuer l'ACP.
from sklearn.preprocessing import StandardScaler
# matplotlib : pour dessiner des graphiques, des tracés ...
import matplotlib.pyplot as plt
```

  2. Importation des données :

```
# importer les données de notre fichier dataset demo.
Nutrient_data = pd.read_csv("./nndb_flat1.csv",sep = ',')

# informations sur les données
Nutrient_data.info()
```

```
class 'pandas.core.frame.DataFrame'>
angeIndex: 8618 entries, 0 to 8617
ata columns (total 45 columns):
#   Column            Non-Null Count  Dtype
--  ------            --------------  -----
0   ID                8618 non-null   int64
1   FoodGroup         8618 non-null   object
2   ShortDescrip      8618 non-null   object
3   Descrip           8618 non-null   object
4   CommonName        1063 non-null   object
5   MfgName           1560 non-null   object
6   ScientificName    732 non-null    object
7   Energy_kcal       8618 non-null   float64
8   Protein_g         8618 non-null   float64
9   Fat_g             8618 non-null   float64
10  Carb_g            8618 non-null   float64
11  Sugar_g           8618 non-null   float64
12  Fiber_g           8618 non-null   float64
13  VitA_mcg          8618 non-null   float64
```

  3. Vérification des données :

```
# La méthode Head renvoie les 5 premières lignes de données
print( Nutrient_data.head() )
```

```
   ID              FoodGroup             ShortDescrip  \
1001  Dairy and Egg Products          BUTTER,WITH SALT
1002  Dairy and Egg Products  BUTTER,WHIPPED,WITH SALT
1003  Dairy and Egg Products      BUTTER OIL,ANHYDROUS
1004  Dairy and Egg Products               CHEESE,BLUE
1005  Dairy and Egg Products              CHEESE,BRICK

                  Descrip CommonName MfgName ScientificName  Energy_kcal  \
            Butter, salted        NaN     NaN            NaN        717.0
Butter, whipped, with salt        NaN     NaN            NaN        717.0
     Butter oil, anhydrous        NaN     NaN            NaN        876.0
              Cheese, blue        NaN     NaN            NaN        353.0
             Cheese, brick        NaN     NaN            NaN        371.0

  Protein_g  Fat_g  ...  Folate_USRDA  Niacin_USRDA  Riboflavin_USRDA  \
       0.85  81.11  ...        0.0075      0.002625          0.026154
       0.85  81.11  ...        0.0075      0.002625          0.026154
       0.28  99.48  ...        0.0000      0.000188          0.003846
      21.40  28.74  ...        0.0900      0.063500          0.293846
```

```
# L'attribut shape renvoie les dimensions de notre Nutrient_data
print( Nutrient_data.shape )

# il y a 8618 échantillons et 45 variables, (Etat : étiquette )
```

```
(8618, 45)
```

4. Prétraitement :

```
# séparation des variables non numériques

# l'index pour les données , inplace(true): modifie le meme objet DataFrame sans créer un autre modifié.
Nutrient_data.set_index('ID', inplace=True)

# sauvegarder les libellés pour les utiliser après
Nutrient_data_Libelles = Nutrient_data.iloc[:,:6]

# supprimer les libellés
Nutrient_data.drop(Nutrient_data.columns[:6].values,axis = 1,inplace = True)

# La méthode Head renvoie les 5 premières lignes de données
print(Nutrient_data.head())
```

```
      Energy_kcal  Protein_g  Fat_g  Carb_g  Sugar_g  Fiber_g  VitA_mcg  \
ID
1001        717.0       0.85  81.11    0.06     0.06      0.0     684.0
1002        717.0       0.85  81.11    0.06     0.06      0.0     684.0
1003        876.0       0.28  99.48    0.00     0.00      0.0     840.0
1004        353.0      21.40  28.74    2.34     0.50      0.0     198.0
1005        371.0      23.24  29.68    2.79     0.51      0.0     292.0

      VitB6_mg  VitB12_mcg  VitC_mg  ...  Folate_USRDA  Niacin_USRDA  \
ID                                   ...
1001     0.003        0.17      0.0  ...        0.0075      0.002625
1002     0.003        0.13      0.0  ...        0.0075      0.002625
1003     0.001        0.01      0.0  ...        0.0000      0.000188
1004     0.166        1.22      0.0  ...        0.0900      0.063500
```

5. Normalisation:

```
# normalisation et centrage des données
# moyenne =0, variance=1, les échantillons doivent être des lignes sinon vous devez les transposer

scaled_Nutrient_data = StandardScaler().fit_transform(Nutrient_data)

# round , 2 chiffres significatifs
print (" moyenne : ", np.round(scaled_Nutrient_data.mean(), 2))
print (" variance : ", np.round(scaled_Nutrient_data.std(), 2))
```

```
moyenne :  0.0
variance :  1.0
```

6. ACP :

```
# instanciation d'un objet ACP à entraîner avec notre jeu de données , nombre de composantes à garder : 10
pca_var = PCA(n_components=10)
pca=pca_var.fit_transform(scaled_Nutrient_data)
```

```
# explained_variance_ratio_ : les valeurs propres des composantes principale. (valeurs propres)
print (pca_var.explained_variance_)

# get_covariance() : calculer les covariances à partir de la matrice de données.
print ( pca_var.get_covariance() )

# explained_variance_ratio_ : array avec les proportions de variance associées aux axes. (combien il porte d'informat

# les 10 premiers vecteurs propres représentent 81% de l'information disponible , ils seront conservés
print ( pca_var.explained_variance_ratio_[:10].sum() )
```

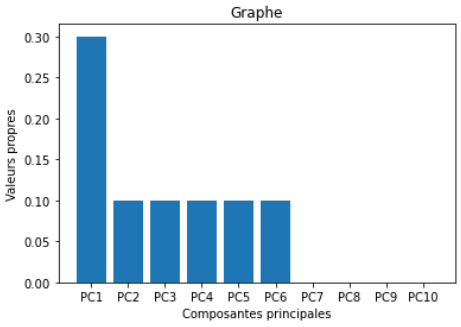
```
[9.86980843 4.11519521 3.42762178 2.93274947 2.3816219  2.09360806
 1.74277356 1.62791371 1.32637001 1.32184789]
[[1.08280312 0.06531234 0.67027476 ... 0.19884722 0.06171755 0.13215807]
 [0.06531234 0.8873928  0.11068698 ... 0.43191203 0.40206038 0.46497594]
 [0.67027476 0.11068698 0.97815793 ... 0.10822164 0.02658201 0.0299106 ]
 ...
 [0.19884722 0.43191203 0.10822164 ... 1.02965137 0.24281829 0.29009234]
 [0.06171755 0.40206038 0.02658201 ... 0.24281829 1.11535044 0.178962  ]
 [0.13215807 0.46497594 0.0299106  ... 0.29009234 0.178962   1.03452459]]
0.8114718823407738
```

7. Graphiques des valeurs propres :

```
# les valeurs propres
per_var = np.round(pca_var.explained_variance_ratio_,decimals=1)

# création des étiquettes pour les composantes principales
labels = ['PC' + str(x) for x in range( 1 , len(per_var) +1 ) ]

# création du graphe
plt.bar( x=range( 1,len(per_var)+1 ), height=per_var, tick_label=labels)
plt.ylabel('Valeurs propres')
plt.xlabel('Composantes principales')
plt.title('Graphe')
plt.show()
```



8. Graphiques des valeurs propres :

```
pca = pd.DataFrame(pca[:, :10], index=Nutrient_data.index)

# ajouter les libellés qu'on a supprimé avant
pca = pca.join(Nutrient_data_Libelle)
pca.rename(columns={0:'c1',1:'c2',2:'c3',3:'c4',4:'c5',5:'c6',6:'c7',7:'c8',8:'c9',9:'c10'}, inplace=True)
```

```
# on voit que les 10 vecteurs sont orthogonaux
np.round(pca.corr(), 10)
```

	c1	c2	c3	c4	c5	c6	c7	c8	c9	c10
c1	1.0	-0.0	-0.0	-0.0	0.0	0.0	-0.0	-0.0	-0.0	-0.0
c2	-0.0	1.0	0.0	0.0	0.0	-0.0	-0.0	-0.0	0.0	-0.0
c3	-0.0	0.0	1.0	-0.0	-0.0	-0.0	-0.0	-0.0	-0.0	-0.0
c4	-0.0	0.0	-0.0	1.0	-0.0	-0.0	0.0	-0.0	-0.0	0.0
c5	0.0	0.0	-0.0	-0.0	1.0	-0.0	0.0	-0.0	0.0	-0.0
c6	0.0	-0.0	-0.0	-0.0	-0.0	1.0	0.0	-0.0	-0.0	0.0
c7	-0.0	-0.0	-0.0	0.0	0.0	0.0	1.0	0.0	-0.0	-0.0
c8	-0.0	-0.0	-0.0	-0.0	-0.0	-0.0	0.0	1.0	-0.0	-0.0
c9	-0.0	0.0	-0.0	-0.0	0.0	-0.0	-0.0	-0.0	1.0	0.0
c10	-0.0	-0.0	-0.0	0.0	-0.0	0.0	-0.0	-0.0	0.0	1.0

9. Interprétation des composantes :

- ### Premier Composant :

```
# copier les 10 composantes dans la variable vects
vects = pca_var.components_[:10]

# pd.Series : Tableau unidimensionnel avec des étiquettes d'axe
one = pd.Series(vects[0], index=Nutrient_data.columns)
one.sort_values(ascending=False)

# ces valeurs representent la contribution des variables initiale à la construction des composantes principales
```

```
Riboflavin_mg       0.262413
Riboflavin_USRDA    0.262413
Niacin_mg           0.258820
Niacin_USRDA        0.258820
VitB6_USRDA         0.240479
VitB6_mg            0.240479
Folate_USRDA        0.212663
Folate_mcg          0.212663
Iron_mg             0.207546
Thiamin_mg          0.204840
Thiamin_USRDA       0.204840
Zinc_USRDA          0.187761
Zinc_mg             0.187761
Magnesium_USRDA     0.166961
Magnesium_mg        0.166961
Phosphorus_mg       0.147768
```

- ### Deuxième Composant :

```
two = pd.Series(vects[1], index=Nutrient_data.columns)
two.sort_values(ascending=False)
```

```
VitB12_mcg          0.359418
VitB12_USRDA        0.359418
VitA_USRDA          0.338184
VitA_mcg            0.338184
Copper_mcg          0.306495
Copper_USRDA        0.306495
Selenium_mcg        0.134056
Selenium_USRDA      0.134056
```

- ## Conclusion :

On constate qu'on a trouvé les mêmes résultats (variabilité cumule = 81,147% et les valeurs propres) que ça soit avec EXCEL ou Python, donc on a atteint l'intérêt de l'ACP et on a arrivé à réduire le nombre des axes de 38 à 10.